A Polynomial Time Approximation Scheme for Dense Min 2Sat

Cristina Bazgan[1] W. Fernandez de la Vega[2]

[1] Université Paris-Sud, LRI, bât.490, F–91405 Orsay, France, bazgan@lri.fr
[2] CNRS, UMR 8623, Université Paris-Sud, LRI, F–91405 Orsay, France, lalo@lri.fr

Abstract. It is proved that everywhere-dense Min 2SAT and everywhere-dense Min Eq both have polynomial time approximation schemes.

1 Introduction

The approximability theory of dense instances of maximization problems such as Max Cut, Max 2SAT has had many recent successes, starting with [1] and [2]. (See [5] for a recent review.) In [1] it is proved in particular that the dense instances of any problem in Max SNP have a *polynomial-time approximation scheme*. In [4], it is proved that many of these problems can be approximated in constant time with an additive error ϵn^2 where n is the size of the input in a certain probe model (implying that the dense versions have constant-time approximation schemes).

The case of dense instances of minimization problems (or edge-deletion problems) seems to be harder. The case of Bisection was settled in [1]. In this paper, we bring a further contribution to this case by proving that everywhere-dense Min 2SAT and everywhere-dense Min Eq both have polynomial-time approximation schemes. Our main tool is a constrained version of Bisection, which we call Paired Bisection: a pairing Π of the vertices is given and we look only at the bisections which split each pair of vertices in Π. The key step in the proof is an L-reduction from Min Eq to Paired Bisection. Then we adapt the algorithm of [1] for Bisection to Paired Bisection. This yields a polynomial-time approximation scheme for everywhere-dense Min Eq. A density preserving L-reduction from Min 2SAT to Min Eq concludes the proof.

2 Preliminaries

We begin with some basic definitions.

Approximability. Let us recall a few definitions about approximability. Given an instance x of an optimization problem A and a feasible solution

y of x, we denote by $m(x, y)$ the value of the solution y, and by $opt_A(x)$ the value of an optimum solution of x. In this paper we consider mainly minimization problems. The *performance ratio* of the solution y for an instance x of a minimization problem A is

$$R(x, y) = \frac{m(x, y)}{opt_A(x)}.$$

For a constant $c > 1$, an algorithm is a *c-approximation* if for any instance x of the problem it returns a solution y such that $R(x, y) \leq c$. We say that an optimization problem is *constant approximable* if, for some $c > 1$, there exists a polynomial-time c-approximation for it. APX is the class of optimization problems that are constant approximable. An optimization problem has a *polynomial-time approximation scheme* (a ptas, for short) if, for every constant $\varepsilon > 0$, there exists a polynomial-time $(1 + \varepsilon)$-approximation for it.

L-reduction. The notion of L-reduction was introduced by Papadimitriou and Yannakakis in [6]. Let A and B be two optimization problems. Then A is said to be *L-reducible* to B if there are two constants $\alpha, \beta > 0$ such that

1. there exists a function, computable in polynomial time, which transforms each instance x of A into an instance x' of B such that $opt_B(x') \leq \alpha \cdot opt_A(x)$,
2. there exists a function, computable in polynomial time, which transforms each solution y' of x' into a solution y of x such that $|m(x, y) - opt_A(x)| \leq \beta \cdot |m(x', y') - opt_B(x')|$.

For us the important property of this reduction is that it preserves ptas's; that is, if A is L-reducible to B and B has a ptas then A has a ptas as well.

Equivalence. Given n variables, an *equivalence* is an expression of the form $l_i \equiv l_j$ where l_i, l_j are literals. The equivalence $l_i \equiv l_j$ is true under an assignment A iff A gives the same truth value (*true* or *false*) to l_i and l_j.

Graphs. As usual, we write $G = (V(G), E(G))$ for the undirected graph with vertex set $V(G)$ and edge set $E(G)$. The vertices are indexed by the integers $1, ..., n = |V(G)|$. For two vertices u and v, uv denotes the edge linking u to v. We denote by $\Gamma(u)$ the set of neighbors of u. If S and T are two disjoint subsets of $V(G)$, we denote by $e(S, T)$ the number of edges linking S to T.

Bisection. Let $G = (V(G), E(G))$ be an undirected graph with an even number of vertices. A *bisection* of G is a partition of vertex set $V(G)$ in two equal size sets R and L. The value of the bisection is the number of edges between R and L.

Paired Bisection. Let $G = (V(G), E(G))$ be a graph with $|V| = 2n$ and let a pairing Π of $V(G)$ be fixed, $\Pi = \{\{u_1, v_1\}, ..., \{u_n, v_n\}\}$, say, (with $\cup_{1 \le i \le n}\{u_i, v_i\} = V(G)$). We say that a bisection $\{R, L\}$ of G is *admissible* with respect to Π, (admissible for short), iff it splits each pair $\{u_i, v_i\}$, (i.e., for $i = 1, \ldots, n$, either $u_i \in R$ and $v_i \in L$ or $v_i \in R$ and $u_i \in L$). We call PAIRED BISECTION the problem of minimizing the value of an admissible bisection where of course Π is part of the data. (See the formal definition below.)

Dense Instances. A graph with n vertices is *δ-dense* if it has at least $\delta n^2/2$ edges. It is *everywhere-δ-dense* if the minimum degree is at least δn. Similarly, a 2CNF formula or a set of equivalences on n variables is *everywhere-δ-dense* if for each variable the total number of occurrences of the variable and its negation is at least δn. A 2CNF formula (a set of equivalences) on n variables is *δ-dense* if the number of clauses (equivalences) is at least δn^2. A set of instances is *dense* if there is a constant $\delta > 0$ such that it is δ-dense and a set of instances is everywhere-dense if there is a constant $\delta > 0$ such that it is everywhere-δ-dense. So, everywhere-dense implies dense but the converse is not true.

We now define the problems in question formally.

MIN 2SAT

Input: A 2CNF formula F.

Solution: A truth assignment for the variables.

Value: The number of clauses satisfied by the assignment.

MIN EQ

Input: A set of equivalences.

Solution: A truth assignment for the variables.

Value: The number of equivalences satisfied by the assignment.

BISECTION

Input: A graph $G = (V(G), E(G))$.

Solution: A bisection of G.

Value: The number of edges in the bisection.

PAIRED BISECTION
Input: A graph $(V(G), E(G))$ with $|V| = 2n$ and a pairing Π of $V(G)$, $\Pi = \{\{u_1, v_1\}, ..., \{u_n, v_n\}\}$.
Solution: A bisection of G which splits each pair $\{u_i, v_i\}$.
Value: The number of edges in the bisection.

All these problems are minimization problems, i.e., a solution with value as small as possible is sought in each case.

3 The Results

Our main result is

Theorem 1. *Everywhere-dense* MIN 2SAT *and everywhere-dense* MIN EQ *both have ptas.*

In the course of proving Theorem 1, we also obtain the next result, which has some interest in view of the fact that the approximability status of BISECTION is wide open.

Theorem 2. PAIRED BISECTION *is APX-hard.*

Remark. It is easy to see that (simply) dense instances of MIN 2SAT or MIN EQ *do not* have a ptas if $P \neq NP$. As far as we know, these are the only problems which are known to have a ptas in the everywhere-dense case but not in the dense case.

The proof of Theorem 1 occupies the rest of the paper. First, we give a density preserving L-reduction from MIN 2SAT to MIN EQ (Lemma 1). As already mentioned, the key step in the proof of Theorem 1 is an L-reduction from MIN EQ to PAIRED BISECTION (Lemma 2). The proof is then easily completed by adapting the ptas for everywhere-dense BISECTION of [1] to obtain a ptas for everywhere-dense PAIRED BISECTION.

4 The Proofs

Lemma 1. *There is an L-reduction from* MIN 2SAT *to* MIN EQ.

Proof. Let F be a set of clauses with at most two literals on n variables $x_1, \ldots, x_n$. We construct a set of equivalences E as follows. We add a new variable y and we replace each clause $l_i \vee l_j$ in F by the following set of equivalences: $l_i \equiv \neg l_j, l_i \equiv \neg y, l_j \equiv \neg y$. By inspection, one sees that if

$l_i \vee l_j$ is satisfied by some assignment, at most 2 of these 3 equivalences are true, so that the inequality $opt(E) \leq 2opt(F)$ holds, showing that the first condition of the definition of the L-reduction is satisfied. Now, suppose that we have a solution of E (an assignment A for the variables that appear in E). We can suppose that $y = false$ in A since the complementary assignment satisfies the same number of equivalences. We consider the same assignment for the variables in F. Let B denote this second assignment. Now one sees that if $l_i \vee l_j$ is satisfied by B then exactly 2 of the equivalences in E corresponding to $l_i \vee l_j$ are satisfied, so that the values satisfy $m(F, B) = m(E, A)/2$, showing that the second condition of the definition of the L-reduction is also satisfied.

Lemma 2. Min Eq *and* Paired Bisection *are mutually L-reducible one to the other.*

Proof. Firstly we construct a L-reduction from Paired Bisection to Min Eq. Let $G = (V(G), E(G))$ be a graph and $\Pi = \{\{u_1, v_1\}, ..., \{u_n, v_n\}\}$ a pairing of $V(G)$. For convenience, we consider that each pair in Π is ordered. We can then represent a bisection (L, R) of G by a vector of n logical variables $\{x_1, ..., x_n\}$ with the understanding that, if $x_i = true$ then we put u_i in L and v_i in R and if $x_i = false$ then we put u_i in R and v_i in L.

Now, for each edge $u_i v_j \in E(G)$ we introduce the equivalence $x_i \equiv x_j$. For each edge $u_i u_j \in E(G)$ or $v_i v_j \in E(G)$ we introduce the equivalence $x_i \equiv \neg x_j$. Call E the set of all these equivalences. By inspection, one can see that an edge of G contributes to the bisection (L, R) exactly when the the corresponding equivalence holds. This implies clearly $opt(E) = opt(G)$ and the L-reduction in one direction.

For the reduction in the other direction, we replace each equivalence $x_i \equiv \neg x_j$ by the edges $u_i u_j$ and $v_i v_j$ and each equivalence $x_i \equiv x_j$ by the edges $u_i v_j$ and $v_i u_j$.

It is straightforward to check that the reductions of Lemma 3 and Lemma 4 map an everywhere-dense set of instances into another everywhere-dense set of instances.

Corollary 1. Paired Bisection *is APX-hard.*

Proof. In [3] it is proved that the following problem is APX-hard: given a set of equivalences find an assignment that minimize the number of equivalences that we have to remove such that the new set of equivalences is satisfiable. There is a simple L-reduction between the above problem and Min EQ (we replace each equivalence $\ell_i \equiv \ell_j$ by $\ell_i \equiv \neg \ell_j$) that implies that Min EQ is APX-hard. The Lemma follows immediately from Lemma 1.

Theorem 3. *Everywhere-dense* Paired Bisection *has a ptas.*

As already mentioned, our ptas is a rather straightforward modification of the ptas of [1] for everywhere-dense Bisection. The main difference is the fact that in our case we don't have to care of the "equal sides" condition which is implicit in the pairing Π, and our algorithm is in fact simpler than that of [1].

Let the input be (G, Π) with $\Pi = \{\{u_1, v_1\}, ..., \{u_n, v_n\}\}$ and $G = (V(G), E(G))$. Let ϵ be the allowed error and $\alpha = \frac{4\delta^2\varepsilon}{25}$. As in [1] we run two distinct algorithms and select the solution with the smallest value. The first algorithm gives a good solution for the instances whose minimum value is at least αn^2 and the second for the instances whose minimum value is less than αn^2.

1. First algorithm (Algorithm for the case of "large" bisection)

Let y_i indicate the side (0 for Left, 1 for Right) of the vertex u_i in the bisection (L, R). [1] use smooth polynomial integer programming for the instances with large optimum value. We just have to check that we can express the value of Paired Bisection as a degree 2 polynomial in the y_i's:

$$\sum a_{ij} y_i y_j + \sum b_i y_i + d$$

where each $|a_{ij}| \leq c, |b_i| \leq cn, |d| \leq cn^2$ for some fixed constant c. We can use

$$\begin{aligned}\text{Paired Bisection} = \min & \sum_{u_i v_j \in E(G)} (y_i(1-y_j) + y_j(1-y_i)) \\ & + \sum_{u_i u_j \in E(G)} [1 - (y_i y_j + (1-y_i)(1-y_j))] \\ & + \sum_{v_i v_j \in E(G)} [1 - (y_i y_j + (1-y_i)(1-y_j))]\end{aligned}$$

This program can be solved approximately in polynomial time by an algorithm of [1].

2. Second algorithm (Algorithm for the case of a "small" bisection)

This second algorithm is again similar to that of [1]. However, it will be seen that important differences appear and we felt the need for a new (albeit sketched) correctness proof although this proofs relies heavily on [1]. Actually, the proof of correctness of the algorithm of [1] for the case of small bisection relies on the property that one can assume that the vertices in one side of the bisection have no negative bias. (Lemma 5.1 in [1]. We define the bias of a vertex u as the difference between the number of edges it sends to his side in the bisection and the number of edges it sends to the other side.) There is apparently no analogue to this property in our case. [1] use exhaustive sampling for the case of a "small" bisection. Here we sample the set of pairs Π rather than the set of vertices. Actually, we will work with pairs all along the way. Let S be the set theoretical union of $m = O((\log n)/\delta)$ pairs picked randomly from Π. We can assume by renaming that $S = \cup_{i=1}^{m}\{u_i, v_i\}$.

Let (L_o, R_o) be an optimal admissible bisection of G and let $S_L = S \cap L_o, S_R = S \cap R_o$. Again by renaming, we can assume that $S_L = \{u_1, ..., u_m\}$ and $S_L = \{v_1, ..., v_m\}$. (Actually, the algorithm which does not know the partition (S_L, S_R), will be run for each of the 2^{m-1} admissible partitions of S.)

As in [1], the placement is done in two stages. In the first stage, pairs of vertices are placed on the basis of their links with S. (An important difference with the algorithm of [1] occurs here: in the algorithm of [1], only "right" vertices are placed at this step.) In the second step, the remaining pairs are placed on the basis of their links with the vertices placed during the first step and with S. In the description below, we let L and R denote the current states of the left-hand side (resp. right-hand side) of the bisection constructed by the algorithm. Thus, we start with $L = S_L,\ R = S_R$.

1. Let

   $$T_1 = \{i > m : |\Gamma(u_i) \cap S_R| + |\Gamma(v_i) \cap S_L| \leq (|\Gamma(u_i) \cap S_L| + |\Gamma(v_i) \cap S_R|)/2\}$$

   $$T_2 = \{i > m : |\Gamma(u_i) \cap S_L| + |\Gamma(v_i) \cap S_R| \leq (|\Gamma(u_i) \cap S_R| + |\Gamma(v_i) \cap S_L|)/2\}$$

   For each $i \in T_1$ we put u_i in L and v_i in R. For each $i \in T_2$ we put u_i in R and v_i in L.
2. Let $L_1 = S_L \cup (\cup_{i \in T_1}\{u_i\}) \cup (\cup_{i \in T_2}\{v_i\})$ denote the set of vertices placed on the left side after the completion of stage 1, and similarly, let $R_1 = S_R \cup (\cup_{i \in T_1}\{v_i\}) \cup (\cup_{i \in T_2}\{u_i\})$ denote the "right" vertices. Let $J = \{m+1, ..., n\} \backslash (T_1 \cup T_2)$. For each $i \in J$
   (a) if $|\Gamma(u_i) \cap R_1| + |\Gamma(v_i) \cap L_1| \leq |\Gamma(u_i) \cap L_1| + |\Gamma(v_i) \cap R_1|$ then we add u_i in L and v_i in R;
   (b) otherwise we add v_i in L and u_i in R.

Let us sketch now a proof of correctness of the second algorithm. (algorithm for "small" bisection). We denote by $opt(G) = opt(G, \Pi)$ the value of an optimum admissible bisection of G.

Lemma 3. *With high probability,*

1. *T_1 contains each index i with the property that $|\Gamma(u_i) \cap R_o| + |\Gamma(v_i) \cap L_o| \leq (|\Gamma(u_i) \cap L_o| + |\Gamma(v_i) \cap R_o|)/4$*
2. *T_2 contains each index i with the property that $|\Gamma(u_i) \cap L_o| + |\Gamma(v_i) \cap R_o| \leq (|\Gamma(u_i) \cap R_o| + |\Gamma(v_i) \cap L_o|)/4$*

Also with high probability, each pair in the set $\{(u_i, v_i) : i \in T_1 \cup T_2\}$ is placed as in the optimum solution.

Proof. The proof is completely similar to that of Lemma 5.2 in [1] and is omitted. We remark in passing that sample size $O(\sqrt{\log n}/\delta)$ suffices (instead of $m = O((\log n)/\delta)$ used in [1]).

Lemma 4. $n - (m + |T_1| + |T_2|) \leq \frac{5\mathrm{opt}(G)}{2\delta n}$.

Proof. The proof of this Lemma is again very similar to the proof of Lemma 5.3 in [1] and is omitted.

Lemma 5. *If $opt(G) < \alpha n^2$ then with high probability the value of the bisection given by algorithm 2 is at least $(1+\varepsilon)opt(G)$ where $\varepsilon = \frac{25\alpha}{4\delta^2}$.*

Proof. We need first some notations. Let $U = \cup_{i \in J}\{u_i, v_i\}$ and let $u = |J|$. (U is the set of vertices which are placed during step 2.) Let $U_L = U \cap L$, $U_R = U \cap R$, $U_L^{opt} = U \cap L_o$ and $U_R^{opt} = U \cap R_o$. Let $m(G, sol)$ denote the value of the bisection given by the algorithm, Let $d(U) = e(U_L, U_R)$ and $d_{opt}(U) = e(U_L^{opt}, U_R^{opt})$. For each $i \in J$, we define

$$val(i) = |\Gamma(v_i) \cap L_1| - |\Gamma(v_i) \cap R_1|$$

if the case (a) of stage 2 of the algorithm occurs for the index i, and

$$val(i) = |\Gamma(u_i) \cap L_1| - |\Gamma(u_i) \cap R_1|$$

otherwise. We denote by d_1 the number of edges of G with exactly one extremity in R_1. Let us check that we have

$$m(G, sol) = d_1 + \sum_{i \in J} val(i) + d(U). \tag{1}$$

Indeed, assume that case (a) occurs for the index i. (The treatment of case (b) is similar.) This means that $u_i \in L,\ v_i \in R$. Then, apart from edges linking U_L to U_R (which are separately counted), $|\Gamma(v_i) \cap L_1|$ new edges contribute to the bisection, and $|\Gamma(v_i) \cap R_1|$ are to be subtracted, since they are counted in d_1 and do not contribute to the bisection.

We see, using Lemma 3, that with high probability, the optimum value of an admissible bisection is

$$\mathrm{opt}(G) = d_1 + \sum_{i \in J} val_{\mathrm{opt}}(i) + d_{\mathrm{opt}}(U)$$

where

$$val_{\mathrm{opt}}(i) = |\Gamma(v_i) \cap L_1| - |\Gamma(v_i) \cap R_1|$$

if $u_i \in U_L^{opt}$ and

$$val_{\mathrm{opt}}(i) = |\Gamma(u_i) \cap L_1| - |\Gamma(u_i) \cap R_1|$$

if $u_i \in U_R^{opt}$. The bisection of U constructed in stage 2 minimizes $\sum_{i \in J} val(i)$. We have thus

$$\sum_{i \in J} val(i) \leq \sum_{i \in J} val_{\mathrm{opt}}(i).$$

This implies, with (1)

$$\begin{aligned} m(G, sol) &\leq d_1 + \sum_{i \in J} val_{\mathrm{opt}}(i) + d_{opt}(U) - d_{opt}(U) + d(U) \\ &= opt(G) - d_{opt}(U) + d(U) \leq opt(G) + d(U) \\ &\leq opt(G) + u^2 \leq opt(G) + \frac{25 opt(G)^2}{4\delta^2 n^2} \\ &\leq opt(G) \left(1 + \frac{25\alpha}{4\delta^2}\right) \end{aligned}$$

using Lemma 8.

The correctness follows now from our choice of α.

5 Open problems

The major open problem is of course the approximability or inapproximability of Bisection. Can our approximation hardness theorem for Paired Bisection help?

The true complexity of approximate Min 2SAT in the dense case is another interesting question. It is known that the case of "large" bisection can be done in constant time (see [4]). Can overall constant time be achieved?

References

1. S. Arora, D. Karger, M. Karpinski, *Polynomial time approximation schemes for dense instances of NP-hard problems*, Proc. of 27th STOC, 1995, 284–293. The full paper will appear in Journal of Computer and System Sciences.
2. W. Fernandez de la Vega, *Max-Cut has a Randomized Approximations Scheme in Dense Graphs*, Random Structures and Algorithms, 8(3) (1996), 187–198.
3. N. Garg, V. V. Vazirani and M. Yannakakis, *Approximate max-flow min-(multi)cut theorems and their applications*, SIAM Journal on Computing 25 (1996), 235–251.
4. O. Goldreich, S. Goldwasser and D. Ron, *Property Testing and its Connection to Learning and Approximation*, Proc. of 37th IEEE FOCS, 1996, 339–348. The full paper has appeared in Journal of the ACM, 45 (4) (1998), 653–750.
5. M. Karpinski, *Polynomial Time Approximation Schemes for Some Dense Instances of NP-Hard Optimization Problems*, , Randomization and Approximation Techniques in Computer Science, LNCS 1269, 1–14.
6. C. Papadimitriou and M. Yannakakis, *Optimization, Approximation and Complexity Classes*, Journal of Computer and System Science 43 (1991), 425–440.